# Semestrální práce z předmětu ICZ
# Kytarová ladička

Jan Ingerle

29. října 2000

## Zadání

Implementujte do DSP TMS320C5x kytarovou ladičku. Pro ověření funkce použijte simulátor pro daný procesor.

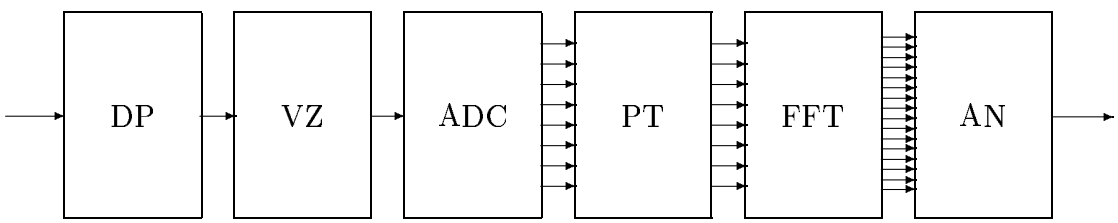
## Řešení

Ladička v podstatě realizuje vybírání harmonického signálu ze směsi. To se dá realizovat několika způsoby:

- výpočtem spektrální výkonové hustoty signálu a její analýzou
- modelováním signálu pomocí AR modelu
- adaptivními metodami

Relativně jednoduchá cesta vede přes výpočet spektrální výkonové hustoty (SPD). Tato metoda je výhodná především z důvodu velké šumové odolnosti. Nevýhodou je použití dlouhého časového úseku k zajištění dostatečné přesnosti. Spokojíme-li se však s menší přesností, pracuje algoritmus spolehlivě.

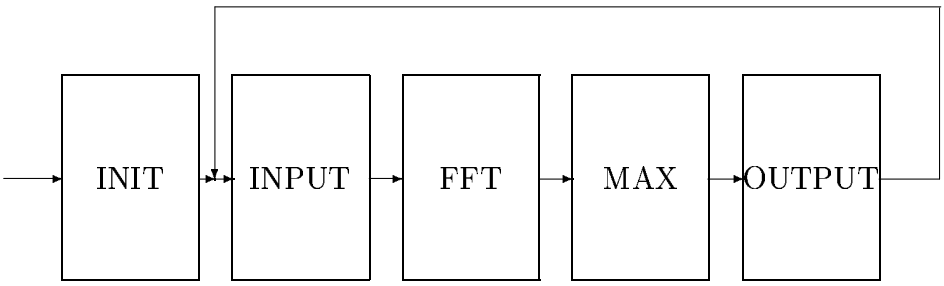
Realizace analýzy pomocí SPD vypadá takto:



Celý řetězec začíná, antialiazingovým filtrem (DP) po kterém následuje vzorkovací obvod (VO) a AD převodník (ADC). Dalším blokem je, potlačení trendu (PT), neboli centrování signálu. Centrování signálu je důležité kvůli prosakování ss. složky ve spektru. Po těchto úpravách následuje rychlá Fourierova transformace (FFT) a analýza PSD (AN).

Při zpracování pomocí signálového signálu se používá kompaktních systémů které obsahují obvody pro předzpracování signálů. Ty realizují filtraci, vzorkování i převod na číslicový signál. Většinou se jako antialiazingový filtr používá pásmová propust, jenž nejen omezuje maximální frekvenci signálu na polovinu vzorkovací frekvence ($f_{max} = \frac{f_s}{2}$), ale odstraňuje i stejnosměrnou složku signálu. Do procesoru tak přichází signál již předzpracovaný a připraven pro vlastní analýzu.

Program procesoru bude provádět následující algoritmus:




Po inicializační fázi procesoru (INIT) začne smyčka tvořená z načítání vstupních vzorků (INPUT), FFT, analýzy PSD (MAX) a výpisu výsledné hodnoty (OUTPUT).

Jednotlivé fáze:

*INIT* — zde se provede nastavení stavových registru procesoru, inicializace periferií, inicializace paměti a nastavení přerušení.

*INPUT* — je sekce, která cyklicky plní paměť vstupními vzorky signálu do doby, než je načteno N vzorku (N je řád FFT). Vzorky jsou čteny sériovým portem z AD převodníku pomocí přerušení. Po nečtení N vzorků je smyčka přerušena a program přejde na dalšího bloku. Vstupní sekce může pracovat jako uzavřená smyčka, neboť není třeba načítat vstupní vzorky spojitě a můžeme si dovolit některé vynechat, aniž by to mělo nějaký vliv na funkci ladičky.

*FFT* — je nejdůležitější a také nejdelší blok programu. Důležitá je jak volba řádu N, tak volba vzorkovací frekvence $f_s$. Protože nemáme na výpočet FFT omezený čas a využíváme pro její výpočet celý výkon procesoru, můžeme volit N co největší. Omezeni jsme pouze pamětí procesoru. V našem případě zvolíme řád N=512. Při volbě vzorkovacího kmitočtu bereme ohled na požadavek co největší přesnosti a smažíme se volit $f_s$ co nejmenší. Frekvence kytarových strun leží v oblasti od 80Hz do 330Hz. Při zachování vzorkovacího teorému budeme volit $f_s = 1000Hz$. Přesnost určení kmitočtu pak bude cca 2Hz.

*MAX* — je blok, jenž vyhledá maximální hodnotu PSD a porovná její polohu s frekvencemi kytary. Frekvence jednotlivých strun kytary jsou: E = 82,41; A = 110,0; d = 146,8; g = 195,9; h = 246,9; $e^1 = 329,6$. Je vidět, že s přesností 2 Hz je ladění jen velice orientační. Aby jsme zamezili zbytečným nepřesnostem vlivem rušení, nebudeme vybírat maximum z celého rozsahu frekvencí, ale omezíme výběr na okno v okolí těchto frekvencí — $f_{min} = 60$Hz a $f_{max} = 400$Hz. Aktuální naměřená frekvence se porovnává s nejbližší frekvencí struny.

*OUTPUT* — poslední blok zapisuje na výstupní paralelní port informaci o aktuální poloze měřené frekvence. Pokud je aktuální frekvence menší než nejbližší frekvence struny, na výstup se zapíše 1, pokud je frekvence vyšší než nejbližší frekvence struny, na výstup se zapíše 2. Naladění indikuje 0. Po vypsání informace na výstup se vrací program k bloku načítání vzorků.

Celý algoritmus zpracování a vyhodnocování jsem nejprve implementoval v programu `MATLAB`. Zde jsem použil vstupní `wav` soubor s vzorkovací frekvencí 44kHz. Proto jsem tuto frekvenci decimoval na 8kHz. Vzhledem k možnostem `MATLABu` se zde nebylo třeba omezovat na nízký řád FFT a proto je zde dosaženo většího rozlišení. Je zde realizována také funkce pro centrování signálu. Celý program je přiložen.

Při tvorbě programu pro DSP jsem použil program pro FFT N=512, firmy Texas Instrument. Celý program kytarové ladičky je přiložen.

# Závěr

S využitím programu pro implementaci FFT se podařilo realizovat požadovaný program. Program jsem otestoval na simulátoru daného procesoru. Při testování jsem použil signál měnící svou frekvencí s krokem 2Hz. Po průchodu programu jsem získal posloupnost obsahující 0,1,2. Při porovnání s vstupním signálem jsem zjistil, že program pracuje správně — ve zvoleném intervalu frekvencí ukazuje zda je aktuální frekvence větší či menší než nejbližší frekvence struny nebo že je s touto frekvencí totožná. Vně tohoto intervalu jsou údaje nepřesné, což se dalo očekávat.

Pokusil jsem se spustit daný program na přípravku s daným signálový procesorem, ale vzhledem k faktu, že FFT 512 zabírá notnou část paměti a v procesoru je umístěn ladící program, nepodařilo se mi program správně alokovat do paměti a rozběhnout.

# Literatura

[1] Davídek,V., Sovka,P.: Číslicové zpracování signálů a implementace, ČVUT, 1996

[2] `http://www.ti.com/`

[3] `http://amber.feld.cvut.cz/psp/`

# Výpis program pro MATLAB

```
ladicka.m
%%%%%%%%

function ladicka(str,fvs,jmeno);

%funkce vypocita frekvenci struny i referencni struny
%a urci rozdil
%
%ladicka(str,fvs,jmeno);
%
%str     ... jmeno promenne se zaznamem struny
%fvs     ... vzorkovaci frekvence struny
%jmeno   ... struna (E,A,d,g,h,e1)
%
%frekvence strun jsou v promene fmr
%cislo zpracovavaneho segmentu je v promene cislodegs
%decimacni pomer je dan pomoci decfrekv
%stupen fir filtru je v promene n
%delka segmentu v sekundach je v promene interval
%frekvencni rozliseni je v promene presnost
%redukce je promenna pro omezeni delky signalu
%intplkon je interpolacni konstanta

%-----------------nastaveni-------------------

switch jmeno
   case 'E', fmr=82.41; %137
   case 'A', fmr=110.0; %184
   case 'd', fmr=146.8; %124
   case 'g', fmr=195.9; %166
   case 'h', fmr=246.9; %206
   case 'e1', fmr=329.6; %274
end;

cislosegs=1;
decfrekv=11025;
n=300;
presnost=0.1;

%interval=1/presnost;
%redukce=interval; % => 1s zaznamu
%intplkon=redukce; % => zacovani presnosti
interval=1;
intplkon=1/presnost;

%-----------------------------------------------
```

```
dec=fvs/decfrekv;

str=decimate(str,dec,n,'fir');
fvs=fvs/dec;

disp('ladim...');

%delkasegm=ceil(fvs*interval/redukce);
delkasegm=ceil(fvs*interval);
str=str((cislosegs-1)*delkasegm+1:cislosegs*delkasegm);

fms=struna(str,fvs,intplkon);

disp(sprintf('Snazis se naladit na frkvenci %.2f.',fmr));
disp(sprintf('Struna ma frkvenci %.2f.',fms));
if fmr<fms
   stav='treba strunu trochu povolit!';
elseif fmr>fms
   stav='treba strunu vice nasponovat!';
else
   stav='uspesne naladeno!';
end;
disp(sprintf('Je tedy jasne, ze je %s',stav));


struna.m
%%%%%%%

function fm=struna(signal,fv,interpol);
%struna(signal,fv);
%
%signal   ... jmeno signalu
%fv       ... vzorkovaci frekvence
%interpol ... konstatnta pro interpolaci
%
%od daneho signalu odecte trend,
%provede interpolaci ve spektru, provede fft a
%zjisti maximum ve frekvencnim spektru v danem intervalu
%
%interval je v promene intvl

%-------------------nastaveni----------------------

intvl=[65 340];

%--------------------------------------------------

delka=length(signal);
tre=trend(signal,fv);
signal=signal-tre;
```

```
nuly=zeros(1,ceil(interpol*delka));
signal=[signal' nuly];
delka=length(signal);

SIGNAL=fft(signal);

[maximum,poloha]=max(abs(SIGNAL(round(intvl(1)*delka/fv):round(intvl(2)*delka/fv))));
fm=fv/delka*poloha-1+intvl(1);
```

trend.m

%%%%%%

```
function tr=trend(signal,fv);

%vypocet trendu v signalu (fm = 45)
%
%tr=trend(signal,fv)
%
%signal ... dany signal
%tr     ... vypocitany trend
%fv     ... vzorkovaci kmitocet
%
%stupen fir filtru je v promene n

%----------------------nastaveni----------------------

n=300;

%-----------------------------------------------------

ms=length(signal);
dec=floor(fv/(2*900));

signal=decimate(signal,dec,n,'fir');

wb=45*2*dec/fv;
b=fir1(n,wb);
signal=filter(b,1,signal);

tr=interp(signal,dec);
mt=length(tr);
if ms<mt
   tr=tr(1:ms);
else
   tr=[tr ones(ms-mt)];
end;
```